# 一是应该用单引号把%#{userName}%括起来，二是不能用#号，而是改成$号，所以正确的配置应该是 and user\_name like '%${userName}%'。

# 看过MyBatis 官方Blog后才知道，远看这样写： <select id="selectStu" primaryType="string" ResultType="student">   SELECT \* FROM student WHERE name LIKE '%${keyword}' </select> 注意正常情况下MyBatis接收参数用#{param} ，而在模糊查询时一定要写成${param}

## spring配置文件：applicationContext.xml

### Java代码

1. <context:annotation-config />
2. <context:property-placeholder location="classpath:timer.properties" /> <!-- 定时器cron表达式的值从配置文件读取，和applicationContext.xml在同一目录下 -->

5. <bean id="timer" class="com.amg.large.core.timer.TimerTask" /> <!-- 执行定时器的主类 -->
6. <task:scheduled-tasks scheduler="myScheduler">
7. <task:scheduled ref="timer" method="test" cron="${cron}"/> <!-- 执行定时器主类的方法名，定时器的cron表达式、设置执行时间 -->
8. </task:scheduled-tasks>
9. <task:scheduler id="myScheduler" pool-size="10"/>

## timer.properties:

1. cron=0 0 1 \* \* ? // 每天凌晨一点执行

主类：TimerTask.java

Java代码  [收藏代码](javascript:void())

1. package com.amg.large.core.timer;
3. import org.slf4j.Logger;
4. import org.slf4j.LoggerFactory;
6. public class TimerTask {
7. private static Logger logger = LoggerFactory.getLogger(TimerTask.class);
8. public void test(){
9. long start = System.nanoTime();
10. CountRecordMain.addCountRecord\_ByTypeIs0();
11. CountRecordMain.addCountRecord\_ByHasTypes2();
12. logger.info("time=" + (System.nanoTime() - start) / Math.pow(10, 9) + "s");
13. }
14. }



# Mybatis中对数据库的简单操作：

、

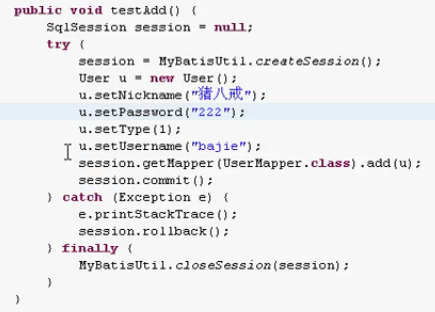


## JUnit测试MyBatis用列：



## 基于Mapper的方法测试用：

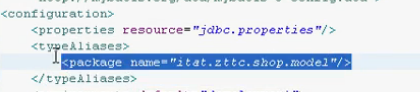
每一个都的创建一个单独的类，建议 不用，使用名称的方法好统一的，





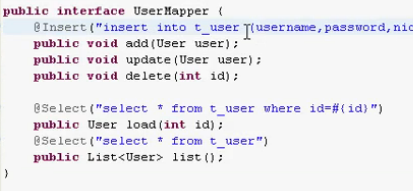
其中用到了类别名来映射；

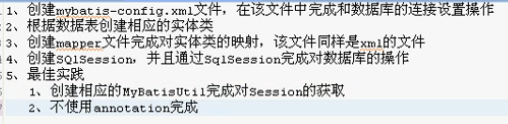




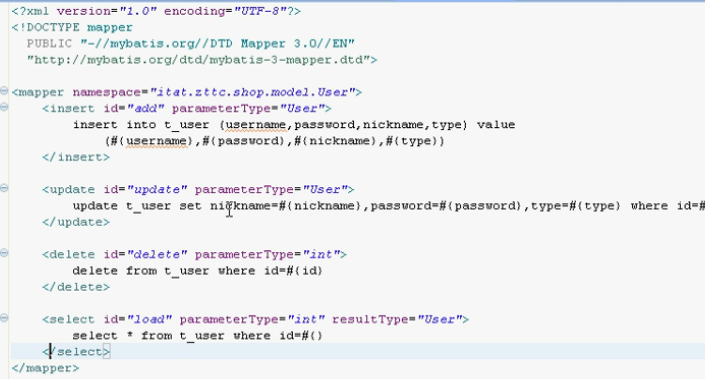
## 加了一个包之后就不需要再添加了；

# 基于注解的写法：就不要引入xml文件，只要引入类就可以实现了；

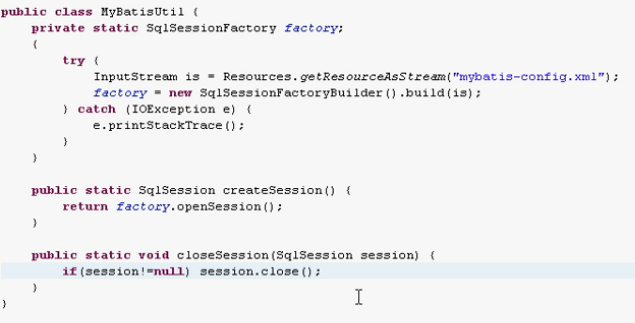


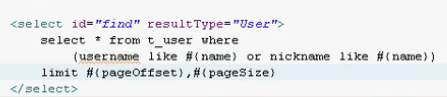


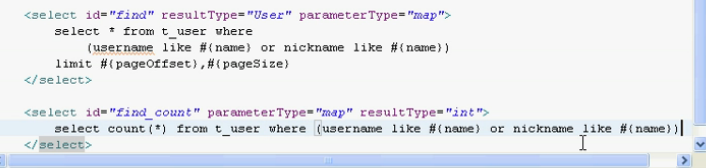
## Sql的映射，建议使用基于sql的访问而不要使用mapper接口处理；



## Mybatis的工具类固定写法如下：

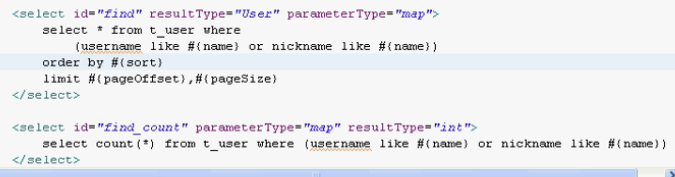




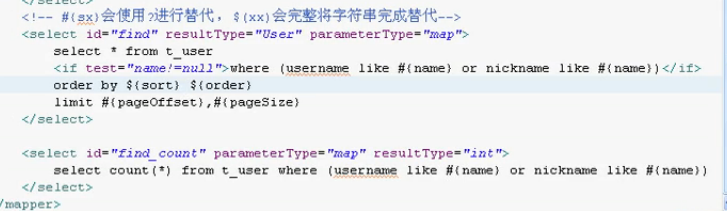


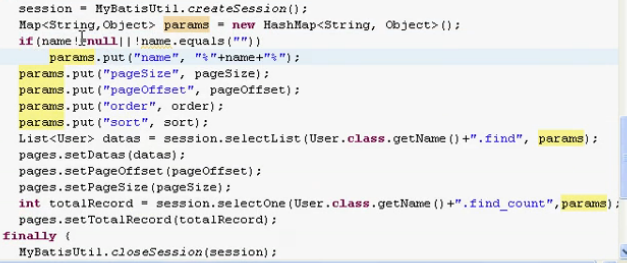
上述查询总条数的方法；





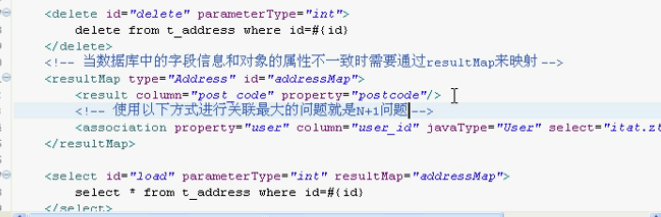
排序方法；





||改为&&；

名字不等于空，才加where条件实现动态sql



## 属性和数据库字段不同的时候，一般select的方式是不用的；

Annotaction注解：



# 可以自己定义一个@名称；